Universidad de Buenos Aires

Facultad de Ingeniería

# Módulo electrónico para laboratorio remoto

## Especificación de Casos de Uso

**Raúl Emilio Romero**
**rraulemilioromero@gmail.com**

**15/11/2023**
**Versión A**

# Registros de cambios

| Revisión | Detalles de los cambios realizados | Fecha |
|---|---|---|
| 0 | Creación del documento | 15/11/23 |

# Contents

# 1 Introducción

## 1.1 Propósito

1. Este documento representa una especificación de casos de uso para un módulo electrónico adquisidor de datos para laboratorios remotos (LR).

2. Está dirigido a desarrolladores que se ocupen del análisis, diseño e implementación, así como también a quienes desarrollen el testing de software.

## 1.2 Definiciones, Acrónimos y Abreviaturas

| | |
|---|---|
| API | Application Programming Interface |
| CEIoT | Carrera de Especialización en Internet de las Cosas |
| CESE | Carrera de Especialización en Sistemas Embebidos |
| IoT | Internet de las Cosas |
| JSON | JavaScript Object Notation |
| MIoT | Maestría en Internet de las Cosas |
| MQTT | Message Queuing Telemetry Transport |
| LR | Laboratorio remoto |
| TBD | To Be Define |
| SACDMELR | Software de adquisición y control de datos para módulos electrónicos de laboratorios remotos |
| UART | Universal Asynchronous Receiver-Transmitter |
| N/A | No Aplica |

## 1.3 Referencias

1. Plan de Proyecto del Trabajo Final de la Carrera Especialización en Internet de las Cosas. Ing. Raúl Emilio Romero. Plan de Proyecto del Trabajo Final.

## 1.4 Visión general del documento

1. Este documento se realiza siguiendo el estándar IEEE Std. 830-1998

# 2 Descripción general del documento

## 2.1 Perspectiva del producto

El plan de trabajo de la MIoT se separó en dos partes; una dedicada al diseño e implementación general del LR y otra al diseño e implementación de los dispositivos de adquisición y control de datos propios de cada experimento (los requerimientos de software de éstos dispositivos se describen en este documento). En ese contexto, durante el desarrollo de la carrera Especialización en Internet de las Cosas (CEIoT) se desarrolló la primera parte del trabajo, la relacionada con los servicios de software e interfaces de usuario. La segunda, se desarrollará en el marco de la carrera Especialización en Sistemas Embebidos (CESE) y apunta al diseño de los dispositivos de adquisición de datos (denominados módulos electrónicos).

La figura 1 muestra un esquema general de la arquitectura del LR desarrollado y las tecnologías intervinientes. Se separa en niveles que van desde los aspectos directamente vinculados al experimento (dispositivo electromecánico) hasta los que se relacionan con la visualización y control del experimento por el usuario. El dispositivo electromecánico se sitúa en el nivel experimental.


Figure 1: Esquema general de la arquitectura del LR.

El nivel de dispositivos lo integrarán módulos electrónicos que trabajan de forma independiente y que manipularán/registrarán variables físicas del experimento. Cada dispositivo deberá contar con los circuitos y elementos electrónicos necesarios para la manipulación de las variables eléctricas del nivel experimental y con la capacidad de comunicarse con el nivel superior de la arquitectura. Esta comunicación se realizará de a través de internet de forma inalámbrica y por medio del protocolo MQTT.

El nivel de procesamiento/persistencia lo integra una API (Application Programming Interface) que se utiliza para gestionar el flujo de información proveniente de los dispositivos, realizar la persistencia de los datos del experimento y de proporcionar un canal de comunicación con el nivel visualización/control. El nivel de visualización/control cuenta con dos interfaces de usuario, una destinada a usuarios estudiantes y otra a administradores del sistema.

En la figura 2 se muestra un esquema general de los componentes que deberá contar cada módulo electrónico. Cada módulo deberá ser capaz de adquirir señales eléctricas provenientes de sensores analógicos, detectar/manipular señales digitales (como interruptores finales de carrera) y controlar motores de corriente continua o paso a paso de baja potencia. Además, se pretende que la configuración básica de cada dispositivo (credenciales de red, alarmas, escalas, habilitar-deshabilitar canales analógicos) se realice por medio de una terminal serie desde una computadora.

## 2.2 Funciones del producto

1. El software aquí especificado brindará las siguientes funcionalidades:

    - Adquisición de datos.

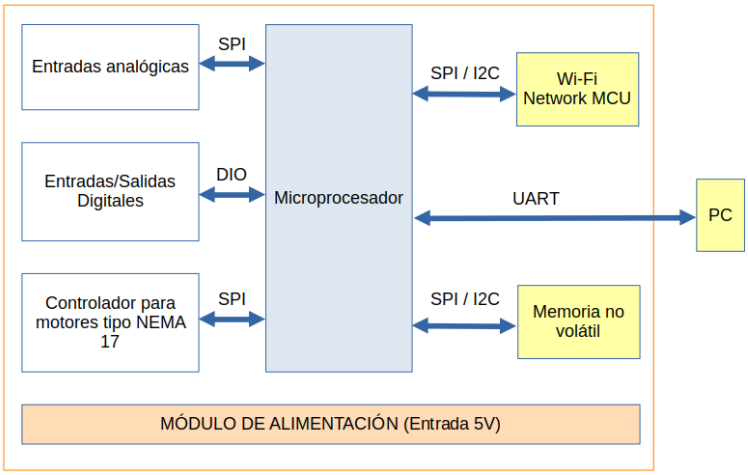


Figure 2: Esquema general del módulo electrónico.

- Control de motores paso-paso de baja potencia.
- Control de motores de corriente continua de baja potencia.
- Transmisión y recepción inalámbrica de datos.
- Comunicación con una PC vía puerto serial.

2. El software aquí especificado no brindará los servicios de:

   - Adquisición y transmisión de datos en tiempo real.

## 2.3 Funciones

### 2.3.1 Configuración

1. El software recibirá vía interfaz serie UART un identificar de dispositivo (ID), almacenará el número en una memoria no volátil, y responderá por medio de un mensaje vía UART que muestre el ID almacenado. [MELR-ER-0001-REQ0010]

2. El software recibirá vía interfaz serie UART las credenciales de una red Wi-Fi, almacenará la configuración en una memoria no volátil, realizará el proceso de conexión a la red y responderá por medio de un mensaje vía UART si pudo conectarse o no a la red. [MELR-ER-0001-REQ0011]

3. El software recibirá vía interfaz serie UART la frecuencia máxima de muestreo por canal analógico a setear, almacenará la configuración en una memoria no volátil y responderá con un mensaje vía UART si la configuración pudo realizarse o no. [MELR-ER-0001-REQ0012]

4. El software recibirá vía interfaz serie UART un mensaje para activar/desactivar el manejo de motores, almacenará la configuración en una memoria no volátil y responderá con un mensaje vía UART si la configuración pudo realizarse o no. [MELR-ER-0001-REQ0013]

5. El software recibirá vía interfaz serie UART un mensaje para seleccionar el tipo de motor a controlar (motor de continua o motor paso-paso), almacenará la configuración en una memoria no volátil y responderá con un mensaje vía UART si la configuración pudo realizarse o no. [MELR-ER-0001-REQ0014]

### 2.3.2 Adquisición y control de datos

1. El software recibirá desde la API del LR un mensaje en formato JSON de acuse de conexión por medio MQTT que deberá responder en el mismo formato con un único campo, ID de dispositivo. [MELR-ER-0001-REQ0015]

2. El software recibirá desde la API del LR un mensaje en formato JSON con la solicitud de adquirir datos analógicos (cantidad de canales y tiempo de registro) y responderá por medio de un mensaje en el mismo formato con los datos adquiridos. [MELR-ER-0001-REQ0016]

3. El software deberá encender o apagar un motor a partir de un mensaje recibido por MQTT en formato JSON desde la API del LR. [MELR-ER-0001-REQ0017]

4. El software deberá cambiar la velocidad (entre dos opciones) de un motor a partir de un mensaje recibido desde MQTT en formato JSON desde la API del LR. [MELR-ER-0001-REQ0018]

5. El software recibirá desde la API del LR un mensaje en formato JSON con la solicitud de los estados de las entradas digitales (estado alto o bajo) y responderá con otro mensaje en el mismo formato con los estados de las entradas digitales. [MELR-ER-0001-REQ0019]

6. El software recibirá desde la API del LR un mensaje en formato JSON con el seteo de los estados de las salidas digitales (estado alto o bajo) y realizará el seteo sobre las salidas. [MELR-ER-0001-REQ0020]

### 2.3.3 Conectividad

1. El software recibirá desde la API del LR un mensaje en formato JSON de solicitud de configuración completa que deberá responder en el mismo formato con los campos: ID, credenciales, frecuencia de muestreo y controlador de motor activado/desactivado). [MELR-ER-0001-REQ0021]

2. El software deberá enviar cada 30 min a la API del LR un mensaje en formato JSON con la siguiente información: ID, nivel de señal Wi-Fi y timestamp). [MELR-ER-0001-REQ0022]

### 2.3.4 Visualización

1. El software encenderá un led de color verde (de forma continua) si esta conectado a una red de Wi-Fi. La condición de no conectado se indicará con un led rojo en modo intermitente (1 segundo de ciclo) y led verde apagado. [MELR-ER-0001-REQ0023]

2. El software encenderá un led de color verde (de forma intermitente durante 3 segundos) cada vez que envíe o reciba un mensaje desde la API de LR. [MELR-ER-0001-REQ0024]

3. El software encenderá un led de color amarillo en modo continuo si el consumo de corriente del módulo electrónico sobrepasa 1 A. Si la corriente es menor a 1 A el led se apagará. [MELR-ER-0001-REQ0025]

# 3 Casos de uso

## 3.1 Configuración de red. [MELR-ECU-0001-CU0001]

| | | |
|---|---|---|
| 1. Nombre | | Configuración de red. [MELR-ECU-CU0001] |
| | 1.1 Breve descripción | En este escenario se cargan las credenciales de una red Wi-Fi para que el módulo pueda conectarse a internet. |
| | 1.2 Actor principal | Técnico electrónico. |
| | 1.3 Disparadores | El técnico electrónico por medio de un terminal serie desde una PC se conecta con el módulo electrónico y envía un mensaje. |
| 2 Flujo de eventos | | |
| | 2.1 Flujo básico | 1. El técnico envía un mensaje vía terminal seríe al módulo con el nombre y contraseña de la red Wi-Fi local.<br>2. El software recibe el nombre de red y contraseña, lo almacena en memoria no volátil y responde por terminal que el dato ha sido almacenado.<br>3. El software realiza la conexión a la red.<br>4. El software responde por terminal que la conexión fue exitosa.<br>5. El software enciende un led verde de forma permanente.<br>6. El software queda en espera de mensajes MQTT. |
| | 2.2 Flujo alternativo | 1. El software no puede realizar la conexión a la red durante un lapso de 2 min.<br>2. El software envía por terminal que la conexión no fue exitosa.<br>3. El software enciende un led rojo.<br>3. El software queda en espera de mensajes por puerto serie. |
| 3. Requerimientos especiales | | N/A |
| 4. Pre-Condiciones | | 1. El módulo electrónico debe estar energizado.<br>2. El switch de configuración debe estar ON.<br>3. El módulo está en modo configuración, es decir, esperar mensajes por terminal serie. |
| 5. Post-Condiciones | | 1. La API del LR debe estar en funcionamineto. |

## 3.2 Adquisición de canales analógicos. [MELR-ECU-0001-CU0002]

| | | |
|---|---|---|
| 1. Nombre | | Adquisición de datos analógicos. [MELR-ECU-CU0002] |
| | 1.1 Breve descripción | En este escenario se solicita desde la API del LR un registro de datos de los canales analógicos. |
| | 1.2 Actor principal | Técnico electrónico. |
| | 1.3 Disparadores | El técnico electrónico por medio de un cliente MQTT envía un mensaje en formato JSON solicitando la adquisición de datos de canales analógicos por un determinado tiempo. El mensaje encapsula: tiempo de registro y cantidad de canales a leer. |
| 2 Flujo de eventos | | |
| | 2.1 Flujo básico | 1. El técnico envía un mensaje JSON por MQTT.<br>2. El software recibe el mensaje.<br>3. El software extrae los datos del mensaje.<br>4. El software realiza el registro de los datos analógicos.<br>5. El software envía de forma periódica (cada 1 segundo) porciones del registro hasta completar la cantidad de muestras correspondientes al tiempo de registro solicitado.<br>6. Terminado el registro, el software envía un mensaje por MQTT acusando el fin del registro.<br>7. El software queda la espera de eventos MQTT. |
| | 2.2 Flujo alternativo | 1. El software no puede interpretar los datos recibidos del mensaje.<br>2. El software envía un mensaje por MQTT informando que no fue capaz de decodificar el mensaje.<br>3. El software pasa al punto 2.1.7 |
| 3. Requerimientos especiales | | N/A |
| 4. Pre-Condiciones | | 1. El módulo electrónico debe estar energizado.<br>2. El switch de configuración debe estar OFF.<br>3. Se debe cumplir el requisito [MELR-ECU-CU0001] |
| 5. Post-Condiciones | | 1. La API del LR debe estar en funcionamineto. |

## 3.3 Encendido de motor. [MELR-ECU-0001-CU0003]

| | |
|---|---|
| 1. Nombre | Encendido de motor. [MELR-ECU-CU0003] |
| 1.1 Breve descripción | En este escenario se solicita desde la API del LR encender un motor a una velocidad determinada. |
| 1.2 Actor principal | Técnico electrónico. |
| 1.3 Disparadores | El técnico electrónico por medio de un cliente MQTT envía un mensaje en formato JSON solicitando el encendido del motor y que gire a una determinada velocidad.<br>El mensaje encapsula la habilitación del motor y la velocidad de giro (entre dos opciones). |
| 2 Flujo de eventos | |
| 2.1 Flujo básico | 1. El técnico envía un mensaje JSON por MQTT.<br>2. El software recibe el mensaje.<br>3. El software extrae los datos del mensaje.<br>4. El software configura el controlador del motor con los parámetros recibidos.<br>5. El software pone en funcionamiento el motor a la velocidad seteada.<br>6. El software a intervalos de 5 min envía un mensaje MQTT informando si el motor sigue encendido.<br>6. El software queda a la espera de eventos MQTT. |
| 2.2 Flujo alternativo | 1. El software detecta que el motor ha permanecido encendido por más de 2 horas.<br>2. El software apaga el motor.<br>3. El software envía un mensaje MQTT informando que se apagó el motor.<br>4. El software pasa al punto 2.1.6 |
| 3. Requerimientos especiales | N/A |
| 4. Pre-Condiciones | 1. El módulo electrónico debe estar energizado.<br>2. El switch de configuración debe estar OFF.<br>3. Se debe cumplir el requisito [MELR-ECU-CU0001]<br>3. El motor debe estar conectado al módulo. |
| 5. Post-Condiciones | 1. La API del LR debe estar en funcionamineto. |

## 3.4 Acuse de conexión. [MELR-ECU-CU0004]

| | | |
|---|---|---|
| 1. Nombre | | Acuse de conexión [MELR-ECU-CU0004] |
| | 1.1 Breve descripción | En este escenario el software envía de forma periódica un log de estado de funcionamiento a la API del LR. |
| | 1.2 Actor principal | Técnico electrónico. |
| | 1.3 Disparadores | Disparo de Timer con ciclo de 30 min. |
| 2 Flujo de eventos | | |
| | 2.1 Flujo básico | 1. El software atiende a una interrupción de un Timer que tiene un ciclo de 30 min.<br>Aclaración: la interrupción se atiende luego de terminar con las tareas que el software esté desarrollado.<br>2. El software encapsula información de ID, configuración de canales analógicos y controlador de motor en un mensaje con formato JSON.<br>3. El software envía vía MQTT el mensaje.<br>4. El software queda a la espera de eventos MQTT. |
| | 2.2 Flujo alternativo | 1. El software no envía el mensaje en el tiempo estipulado.<br>2. El técnico realiza una petición por MQTT de estado de dispositivo. |
| 3. Requerimientos especiales | | N/A |
| 4. Pre-Condiciones | | 1. El módulo electrónico debe estar energizado.<br>2. El switch de configuración debe estar OFF.<br>3. Se debe cumplir el requisito [MELR-ECU-CU0001] |
| 5. Post-Condiciones | | 1. La API del LR debe estar en funcionamineto. |

# 4 Apéndices

N/A.